Report from the Inter-Agency Workshop on HPC Resilience

John T. Daly

Workshop overview

- 3½ day workshop focused on technical challenges of *coordinated* HPC resilience strategies in the exascale timeframe
- 30 participants: system hardware, system software, application developers and users, algorithms and libraries, file systems, I/O and storage, visualization and data analytics
- Highly interactive format
  - problem solving teams of not more than 10 persons each
  - collaboratively develop a plan and roadmap for implementing resilience
- Assessed current capabilities, gaps and dependencies for representative applications from "predictive science" (e.g., PDE) and "not predictive science" (e.g., big data) domains
- Created "proof of concept" resilience strategies and an R&D roadmap for a coordinated resilience framework

Goals of the workshop

- Resilience is about keeping the application workload running to a correct solution in a timely and efficient manner in spite of frequent hard (i.e., unrecoverable) and soft (i.e., recoverable) errors
- Demonstrate both the need for and existence of practical resilience strategies that address the future needs of the application via the resources available to the system
- Create a technical strategy and roadmap for addressing resilience in the exascale timeframe
  - What are our requirements? What "needs" to work and what can we do without?
  - How are we going to meet those requirements?
- Goals of the workshop DO NOT include…
  - Discussion of our current projects or research interests
  - Creating new research opportunities
  - Figuring out how to get funding for future work

Programmatic considerations

- Resilience strategy should not be minimalist or ad hoc, but neither can we afford a "Cadillac" solution
- If we cannot come up with a credible plan for how we intend to integrate all the pieces then mission is likely to view resources expended on resilience as wasted
- If we ask for the kitchen sink we will end up with nothing; if we short-sell we still end up with nothing ➔ not too small; not too big; just right!
- We need…
  - undistracted focus on a "no frills" yet "robust" resilience strategy
  - clear delineation between *engineering* (what we know how to do) and *research* (what we need to figure out)
- Apply Occam's Razor ➔ simplest is best
- I don't care if it's an interesting research question; all I care is if I need to figure it out in order to implement a resilience strategy!

Sample “marching orders”

- Working group sessions will be a single assignment divided somewhat arbitrarily into two assignments
- Make as much progress as you can on every part of the assignment; don’t get stuck – identify open issues; move on
- Start with a simple model for resilience; as time, expertise and resources permit then refine your solutions
- Resilience loves anecdotes and anecdotes are all about the edge cases and extremes ➔ use best engineering judgment to focus on middle of the distribution and not the tails
- Focus on high importance items and “low hanging fruit”; do NOT get distracted by “bright shiny objects” (e.g., better ECC)
- Make it in the context of stated (previous day) application requirements
  - Convert more hard errors into soft errors
  - Provide for reliable and unreliable execution regions
  - Empower the application to make some decisions, without negatively impacting the aggregate system workload

Summary of workshop conclusions

- The number of errors, particularly soft errors, occurring on HPC systems will continue to increase
- A right-sized, well-conceived resilience strategy in the exascale timeframe is more cost effective than continuing to rely on ad-hoc resilience solutions
- Must at a minimum provide for a resilience infrastructure that facilitates
  - System management of hard errors by effectively "converting" them to soft errors whenever feasible
  - Application management of soft errors through interfaces that allow it simple controls over how and when to respond to errors
- Such a framework is foundational to a deployable and sustainable HPC resilience strategy in the exascale timeframe
- Priorities for R&D in the exascale timeframe: fault characterization, detection, FT algorithms, FT programming models and tools

Priority: Fault characterization

- Reliability will get worse with deeply scaled process technologies creating new modes of failure
- Based on anticipated technology trends, the HPC community needs to develop a useful taxonomy for describing
  - the types of faults that future systems are expected to encounter
  - their anticipated frequency and impact

Priority: Detection

- In the exascale timeframe, error “recovery” will likely be manageable using known techniques for local checkpointing
- Fault “prediction” is probably too hard a problem to realistically tackle in this timeframe
- The research focus needs to be error “detection” which requires the system and application to work together in a coordinated fashion
- Industry is not going to solve this problem for the HPC community

Priority: FT Algorithms

- Three classes of algorithms were identified
  a) those that are embarrassingly fault-tolerant
  b) those that are not fault-tolerant but are self-checking
  c) those that are neither fault-tolerant nor self-checking
- Most algorithms currently in class (c) could be moved to class (b) or even class (a) by a moderate R&D investment

Priotity: FT programming models

- Resilience would benefit strongly from a programming model that supports some notion of
  - transactions in time (e.g., roll-back and recovery)
  - transactions in space (e.g., fault containment domains)
- An uncomplicated, directive-based interface using a handful of assertions (e.g., create persistent memory domains, allocate “reliable” and “unreliable” code regions, etc.) provides most of the necessary interfaces for implementing application fault-tolerance

Priority: Tools

- Resilience lacks a mature, validated test infrastructure to verify the effectiveness of various resilience strategies for keeping the application running in the face of high rates of hard and soft errors
- Fault injection tools are need in particular to simulate all classes of faults
- Models will be required to support the fault testing infrastructure at scale